Maximum Bipartite Flow in Networks with Adaptive Channel Width

Yossi Azar[1], Aleksander Mądry[2] ⋆, Thomas Moscibroda[3], Debmalya Panigrahi[2] ⋆⋆, and Aravind Srinivasan[4] ⋆⋆⋆

[1] Microsoft Research, Redmond, WA 98052 and Tel Aviv University, Tel Aviv, Israel
azar@tau.ac.il

[2] Massachusetts Institute of Technology, Cambridge, MA 02139
{madry, debmalya}@mit.edu

[3] Microsoft Research, Redmond, WA 98052
moscitho@microsoft.com

[4] Dept. of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742
srin@cs.umd.edu

Abstract. Traditionally, combinatorial optimization problems (such as maximum flow, maximum matching, etc.) have been studied for networks where each link has a fixed capacity. Recent research in wireless networking has shown that it is possible to design networks where the capacity of the links can be changed *adaptively* to suit the needs of specific applications. In particular, one gets a choice of having *few* high capacity outgoing links or *many* low capacity ones at any node of the network. This motivates us to have a re-look at the traditional combinatorial optimization problems and design algorithms to solve them in this new framework. In particular, we consider the problem of *maximum bipartite flow*, which has been studied extensively in the traditional network model. One of the motivations for studying this problem arises from the need to maximize the throughput of an infrastructure wireless network comprising base-stations (one set of vertices in the bipartition) and clients (the other set of vertices in the bipartition). We show that this problem has a significantly different combinatorial structure in this new network model from the classical one. While there are several polynomial time algorithms solving the maximum bipartite flow problem in traditional networks, we show that the problem is NP-hard in the new model. In fact, our proof extends to showing that the problem is APX-hard. We complement our lower bound by giving two algorithms for solving the problem approximately. The first algorithm is deterministic and achieves an approximation factor of $O(\log N)$, where there are N nodes in the network, while the second algorithm (which is our main contribution) is randomized and achieves an approximation factor of $\frac{e}{e-1}$.

⋆ Research supported by a Fulbright Science and Technology Award, by NSF contract CCF-0829878 and by ONR grant N00014-05-1-0148.

⋆⋆ Part of this work was done while visiting Microsoft Research, Redmond. Research supported in part by NSF contract CCF-0635286.

⋆⋆⋆ Part of this work was done while visiting Microsoft Research, Redmond. Research supported in part by NSF ITR Award CNS-0426683 and NSF Award CNS-0626636.

1 Introduction

Combinatorial optimization has been an important tool in the analysis of computer networks. Traditionally, optimization problems such as maximum flow, maximum matching, etc. have been studied for networks where each link has a fixed capacity.[5] However, recent research in wireless networking has shown that it is possible to design networks where the capacity of the links can be changed *adaptively* to suit the needs of specific applications [4][19]. (We call this phenomenon *adaptive channel width.*) Specifically, wider communication channels increases channel capacity (a result that is also predicted by Shannon's capacity formula [20]), but reduces the transmission range thereby disconnecting distant nodes. Thus, one gets a choice of having *few* high capacity outgoing links or *many* low capacity ones at each node of the network. This motivates us to have a relook at traditional combinatorial optimization problems and design algorithms to solve them in this new framework. In this paper, we focus on the *maximum bipartite flow* problem in networks with adaptive channel width.

Consider a directed bipartite graph on vertices $X \cup Y$, where all the edges are directed from vertices in X to vertices in Y and have capacities. In addition each node in X and Y has a node capacity (corresponding to a budget for nodes in X and demand for nodes in Y). This is equivalent to adding a vertex s (called the *supersource*) which is connected by a directed edge[6] to each vertex in X with edge capacity corresponding to the node capacity; similarly, adding a vertex t (called the *supersink*) such that each vertex in Y is connected to it by a directed edge with the corresponding capacity. In the traditional model, the maximum flow permitted on an edge is equal to its capacity. The *maximum bipartite flow* problem then requires us to find the maximum flow from s to t subject to the capacity constraints. On the other hand, in networks with adaptive channel width, the edges from X to Y do not have fixed capacities; rather one needs to determine the *assigned capacity* at each node $x \in X$. Choosing a high assigned capacity disconnects x from some nodes in Y but produces a high capacity edge to the remaining nodes, while choosing a low assigned capacity implies that all edges from x to nodes in Y have low capacity. The goal is to choose assigned capacities such that the achievable flow in the resulting capacitated network is maximized.

The main practical motivation for studying this problem comes from the need to maximize throughput in infrastructure wireless networks. In particular, let X be the set of base-stations and Y be the set of clients in an infrastructure wireless network. Suppose that the base-stations are equipped with the ability to adapt channel width. We will show that the problem of maximizing throughput in such an infrastructure wireless network is identical to solving the maximum flow problem on bipartite graphs with adaptive channel width. We formally define our throughput maximization problem and its connection to the maximum bipartite flow problem below.

[5] In this paper, we will use the terms *bandwidth* and *capacity* interchangeably.

[6] In this paper, we will use the terms *directed edge*, *link* and *arc* interchangeably.

Problem Definition. We are given a set of base-stations $\mathcal{B}$ and a set of clients $\mathcal{C}$ with $|\mathcal{B}| = n$ and $|\mathcal{C}| = m$. Each base-station $B \in \mathcal{B}$ has a *budget* $\beta(B)$, which is the total capacity that the base-station can deliver to its clients. On the other hand, each client $C \in \mathcal{C}$ has a *demand* $\alpha(C)$, which is the total bandwidth it would like to be allocated from all the base-stations together.

For each base-station and client pair (henceforth, called a *base-client pair*) (B, C), there is a *critical capacity* $\eta(B, C)$, which corresponds to the maximum bandwidth of a link from B to C. To each base station $B \in \mathcal{B}$, the algorithm selects an *assigned capacity* $\tau(B)$ that determines the *capacity* of a link (B, C) (denoted by $\psi_\tau(B, C)$) as follows

$$\psi_\tau(B, C) := \begin{cases} \tau(B) & , \tau(B) \leq \eta(B, C) \\ 0 & , \text{otherwise.} \end{cases}$$

Once the capacities of all links have been fixed, we want to find a *flow* $f(B, C)$ for each link (B, C) such that neither any link capacity is violated (*capacity constraint*), i.e.

$$f(B, C) \leq \psi_\tau(B, C),$$

nor any base-station budget is violated (*budget constraint*), i.e.

$$\sum_{C \in \mathcal{C}} f(B, C) \leq \beta(B).$$

The goal is to find the capacity assignment τ, and corresponding flow f that maximizes the sum of *satisfied demands* of all the clients, where the satisfied demand $\alpha_{\tau,f}(C)$ of a client C is given by

$$\alpha_{\tau,f}(C) = \min\big(\sum_{B \in \mathcal{B}} f(B, C), \alpha(C)\big).$$

Note that given any τ and f, there always exists a flow f' which satisfies the budget and capacity constraints, achieves the same value of total satisfied demand and additionally satisfies the following *demand constraints*,

$$\sum_{B \in \mathcal{B}} f(B, C) \leq \alpha(C).$$

As a result, we will focus on flows that obey demand constraints along with budget and capacity constraints.

The benefit of this assumption is that our problem now corresponds to the maximum bipartite flow problem in networks with adaptive channel width. Recall that in this flow problem, we have two sets of nodes X and Y with edges directed from X to Y, along with a supersource s and a supersink t. To draw the correspondence, let X be the set of base-stations and Y the set of clients. The edge from s to any $x \in X$ (called a *budget arc*) has capacity $\beta(x)$, that from any $x \in X$ to any $y \in Y$ has critical capacity $\eta(x, y)$ and that from any $y \in Y$ to t (called a *demand arc*) has capacity $\alpha(y)$ (refer to Figure 1). We call this graph the *augmentation graph* of the given problem instance. Our task is to choose assigned capacities (i.e. the function τ) for vertices in X; this fixes the capacities of all the arcs from X to Y. Our goal is to choose τ so that the maximum flow in the resulting capacitated network is maximized.

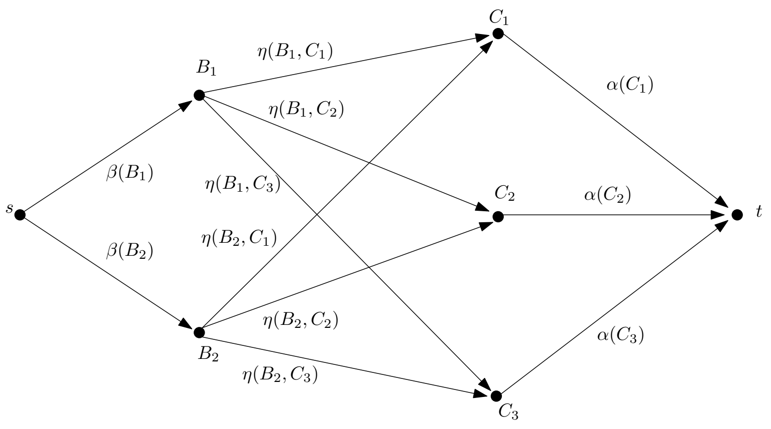

Fig. 1. The *augmentation graph* corresponding to an instance of the problem.

Related Work. Classically, the maximum bipartite flow problem has been solved as a special case of the more general maximum flow problem on arbitrary graphs. Suppose the input graph $G = (V, E)$ has maximum flow of c from the source to the sink. Ford and Fulkerson gave the first algorithm for the maximum flow problem in the 1950s, which had a running time of $O(|E|c)$ [9]. Since then, several algorithms have been developed with better time bounds [6][7][8][11] finally culminating in an $\tilde{O}(|E| \min(|E|^{1/2}, |V|^{2/3}) \log c)$ algorithm due to Goldberg and Rao [10], which is currently the fastest known deterministic algorithm for maximum flow. It may be noted here that a substantial amount of work has also been done for developing randomized algorithms for maximum flow [14][16][15][17], but these algorithms apply only to undirected networks. On the other hand, the maximum bipartite flow problem with unit capacities (which is equivalent to *maximum bipartite matching*) can be solved in $O(|E|\sqrt{|V|})$ time [13].

To summarize the above discussion, the maximum bipartite flow problem in directed graphs with capacitated edges is solvable in polynomial time; however, there is no algorithm which solves this problem faster than in general directed graphs. As we will see, this is in sharp contrast to what we observe in networks with adaptive channel width. In such networks, the maximum bipartite flow problem is NP-hard (in fact, it is APX-hard); further, we give a randomized approximation algorithm achieving an approximation factor of $\frac{e}{e-1}$ which does not appear to extend easily to general directed networks.

We also briefly mention a related class of well-studied problems, namely *unsplittable* flow problems. In these problems, typically there are one or more pairs of source and sink vertices with specific demands, and the goal is to connect the source-sink pairs using paths such that the satisfied demand is maximized while not violating any capacity constraint. These problems are typically NP-hard, and several variants have been studied extensively [18][21][12] [2][3][5]. Interestingly, though we have a single source and a single sink, and flow is allowed to re-distribute arbitrarily at a node, the techniques we use to give an approximation algorithm for our problem bear similarities with the techniques usually used for solving unsplittable flow problems. Specifically, both problems use a suitable linear programming relaxation which is then rounded ensuring that certain cuts are large in the rounded solution.

Our Results. Our first claim is that the maximum bipartite flow problem in networks with adaptive channel width is APX-hard, i.e., it is unlikely that a polynomial-time algorithm can approximate the problem within a certain constant factor. Specifically, we describe an L-reduction from the APX-hard Maximum Bounded 3-Dimensional Matching problem (Max-3DM) to the channel width assignment problem. We prove the following theorem (details of the construction are omitted due to space constraints).

Theorem 1. *The maximum bipartite flow problem in networks with adaptive channel width is APX-hard.*

Our next contribution is a greedy combinatorial algorithm which achieves an approximation factor of $O(\log N)$, where $N = \max(m, n)$. The algorithm first categorizes links according to their critical capacity in geometrically spaced intervals. Now, observe that for any interval, we can set the assigned capacities at the nodes such that all the links in that interval have capacity equal to their critical capacities (while all other links have potentially no capacity at all). The algorithm needs to decide which interval to choose. For this purpose, a maximum flow algorithm is run on the entire graph, assuming that each link has capacity equal to its critical capacity. This outputs a flow on each link. The algorithm greedily chooses the interval which carries the greatest amount of flow on its links. The details of the algorithm are omitted due to space constraints.

Finally, our main result is a randomized algorithm for this problem.

Theorem 2. *There is a randomized algorithm for the maximum bipartite flow problem in networks with adaptive channel width that has an expected approximation factor of* $\frac{e}{e-1}$.

Our algorithm uses a linear programming relaxation of the problem. Recall that the celebrated Menger's theorem implies that maximum flow from s to t equals the minimum $s-t$ cut. So, an algorithm for the problem should aim to choose assigned capacities so as to maximize the minimum $s-t$ cut in the resulting capacitated network. Now, let us consider any linear programming formulation of the problem; such a fractional linear program can be interpreted as a polytope, where its optimal solution is a convex combination of the vertices of the polytope. Each vertex of the polytope represents a particular choice of assigned capacities and therefore, a particular capacitated graph (call them *vertex graphs*); these correspond to the integral solutions we will round our solution to. The natural linear program that we consider first simply ensures that for each cut, the convex combination of the values of the cut in the vertex graphs is large. However, since each vertex graph may have a different minimum $s-t$ cut, this does not guarantee that sizes of these minimum $s-t$ cuts are large. In fact, this linear program has an integrality gap of $\Omega(\log N / \log\log N)$. To overcome this problem, we design a more sophisticated linear program and a corresponding randomized rounding technique that ensures that the minimal $s-t$ cuts in the vertex graphs are large.

2 Algorithms for Maximum Bipartite Flow

As mentioned previously, a greedy combinatorial algorithm for our problem achieves an approximation factor of $O(\log N)$, where $N = \max(m, n)$.

2.1 A Linear Program

Our goal now is to improve upon this combinatorial algorithm. Without loss of generality, we may assume that the assigned capacity chosen at any base-station B in an optimal solution is one among the critical capacities of its outgoing edges, i.e. $\tau(B) \in \{\eta(B,C) : C \in \mathcal{C}\}$. If this is not the case, then the assigned capacity can be increased to the closest value from the set $\{\eta(B,C) : C \in \mathcal{C}\}$ without changing the flow on any link. This allows us introduce the *boolean capacity choice function* $p(B,C)$, which is 1 if $\tau(B) = \eta(B,C)$, and 0 otherwise. Clearly, for any base-station B, $p(B,C) = 1$ for exactly one client C (called the *choice constraint*). We also introduce another new notation, $\mathcal{C}_B(C)$ which represents the set of clients for which the critical capacity of their link to base-station B is less than that for client C, i.e.

$$\mathcal{C}_B(C) = \{C' \in \mathcal{C} : \eta(B,C') \leq \eta(B,C)\}.$$

A natural formulation of the problem is via the following integer linear program (ILP), where constraints (1), (2), (3) and (4) correspond to budget, demand, capacity and choice constraints respectively.

$$\texttt{maximize } \textstyle\sum_{B\in\mathcal{B}}\sum_{C\in\mathcal{C}} f(B,C) \texttt{ subject to}$$

$$\sum_{C\in\mathcal{C}} f(B,C) \leq \beta(B), \;\; \forall B \in \mathcal{B} \tag{1}$$

$$\sum_{B\in\mathcal{B}} f(B,C) \leq \alpha(C), \;\; \forall C \in \mathcal{C} \tag{2}$$

$$f(B,C) \leq \sum_{C'\in\mathcal{C}_B(C)} p(B,C')\eta(B,C'), \;\; \forall B \in \mathcal{B}, \;\; \forall C \in \mathcal{C} \tag{3}$$

$$\sum_{C\in\mathcal{C}} p(B,C) = 1, \;\; \forall B \in \mathcal{B} \tag{4}$$

$$p(B,C) \in \{0,1\}, \;\; \forall B \in \mathcal{B}, \;\; \forall C \in \mathcal{C} \tag{5}$$

$$f(B,C) \geq 0, \;\; \forall B \in \mathcal{B}, \;\; \forall C \in \mathcal{C}. \tag{6}$$

LP Relaxation. To make this ILP tractable, we relax constraint (5) and allow the function p to assume values between 0 and 1 (call this the *fractional program* or FLP). The natural interpretation is that $p(B, C)$ denotes the *goodness* of $\eta(B, C)$ as the assigned capacity of base-station B. Mathematically, it can be thought of as the probability with which $\eta(B, C)$ should be the assigned capacity of base-station B.

Unfortunately, it turns out that this natural linear programming relaxation fails to provide us with an approximation guarantee that is significantly better than the one achieved by the combinatorial algorithm. To understand why this is the case, let us note that for a given choice of values of $p(B, C)$, this linear program is solving the max-flow problem in the augmentation graph where the capacity u of a link (B, C) is the following: if the assigned capacity $\tau(B)$ at base-station B is chosen according to the probability distribution given by $p(B, C)$, then

$$u = \mathbb{E}[\psi_\tau(B, C)].$$

Therefore, by max-flow/min-cut duality, the approach used in the LP boils down to choosing $p(B, C)$ in such a way that the minimal expected capacity among all s-t-cuts in the augmented graph is maximized. Given some final choice of $p(B, C)$ computed by the linear program, it is tempting to round it by choosing assigned capacities according to $p(B, C)$, and then solving the max-flow problem in the resulting graph, hoping that the capacity of minimal cut will be close to the expected one. Clearly, when we focus on one particular cut, say the one that separates base-stations from clients, it will be true, but this does not necessarily mean that for all cuts, such a promise will hold simultaneously. It may happen that for the choice of assigned capacities that we obtain, it will always be the case that for part of the clients the capacity of links leading to them in the resulting graph will be much below the expectation, while for the other part it will excessively large, and this excess will be wasted due to the bottlenecks imposed by not large enough capacity of demand arcs for the respective clients. Thus, even if on expectation each client has reasonable capacity of links leading to it, the rounding procedure might not provide us with a particularly good solution. Therefore, our analysis of the approximation guarantee given by this LP would need to argue that with good probability all cuts are preserved up to some ratio, and in fact, using Chernoff bounds, we can prove that this is indeed true for the ratio $O(\log(m + n)/\log\log(m + n))$. Unfortunately, we can show through an integrality gap example (omitted due to space constraints) that this unsatisfactorily large ratio is not only a shortcoming of our particular rounding procedure, but it is in fact all that we can achieve through *any* rounding algorithm for this LP.

2.2 An Alternative Linear Program

In this section, we will describe a more sophisticated ILP which overcomes the shortcomings of the previous ILP. Note that our goal is to choose assigned capacities such that the augmentation graph has a large maximum flow, or equivalently by Menger's theorem, a large minimum cut. The previous FLP ensures that each cut has large capacity in expectation and therefore the minimum among the expected capacities of the cuts is large; this however does not guarantee that the expected capacity of the minimum cut is large. We need this stronger guarantee from our LP. To achieve this goal, we design an LP which yields a family of flows corresponding to the different choices of assigned capacities, and ensures that the expected value of these flows is large. This clearly implies that the expected capacity of the minimum cut is large, and therefore provides stronger guarantees than the previous LP. Precisely, we consider the following ILP.

$$\texttt{maximize } \textstyle\sum_{B\in\mathcal{B}}\sum_{C',C\in\mathcal{C}} f_{C'}(B,C) \texttt{ subject to}$$

$$\sum_{C\in\mathcal{C}} f_{C'}(B,C) \le p(B,C')\beta(B),\ \ \forall B\in\mathcal{B},\ \ \forall C'\in\mathcal{C} \tag{7}$$

$$\sum_{B\in\mathcal{B}}\sum_{C'\in\mathcal{C}} f_{C'}(B,C) \le \alpha(C),\ \ \forall C\in\mathcal{C} \tag{8}$$

$$f_{C'}(B,C) \le \begin{cases} 0, & \text{if } \eta(B,C) < \eta(B,C') \\ p(B,C')\min\{\eta(B,C'),\alpha(C)\}, & \text{otherwise} \end{cases}$$
$$\forall B\in\mathcal{B},\ \ \forall C,C'\in\mathcal{C} \tag{9}$$

$$\sum_{C\in\mathcal{C}} p(B,C) = 1,\ \ \forall B\in\mathcal{B} \tag{10}$$

$$p(B,C)\in\{0,1\},\ \ \forall B\in\mathcal{B},\ \ \forall C\in\mathcal{C} \tag{11}$$

$$f_{C'}(B,C)\ge 0,\ \ \forall B\in\mathcal{B},\ \ \forall C,C'\in\mathcal{C}. \tag{12}$$

The key to understanding this ILP is the rounding technique that we employ in our approximation algorithm; so let us describe our algorithm first. We relax the integrality constraint , i.e. constraint (11) and allow the variables p to take any value between 0 and 1, both inclusive. We solve the resulting FLP, and then round the solution to obtain an integral solution. It is in this rounding procedure that the crux of our algorithm lies. We choose assigned capacities according to $p(B,C)$, noting that for a fixed base-station B, $p(B,C)$ is a valid probability distribution. Now, for any base-station B, if the assigned capacity $\tau(B)=\eta(B,C')$, then for each link (B,C), we add a flow of $g_{C'}(B,C)\equiv f_{C'}(B,C)/p(B,C')$ to the $s-B-C-t$ path in the augmentation graph. Crucially, this does not violate the budget constraint at any base-station B since the total outflow at B is $\sum_{C\in\mathcal{C}} g_{C'}(B,C)$, which is at most $\beta(B)$ by constraint (7); neither does it violate the capacity constraint on any link (B,C) since constraint (9) ensures that the flow on link (B,C) is at most $\psi_\tau(B,C)$. Hence, we focus on analyzing violations of the demand constraints. The total inflow at C is $\sum_{B\in\mathcal{B}} g_{C'(B)}(B,C)$. Unfortunately, assigning these flow values simultaneously for all base-stations might lead to an overflow in a demand arc (i.e., a demand constraint violation). For a client with overflow, we decrease the incoming flows arbitrarily until the flow on the link to t exactly matches its capacity (we call this the *truncation step*). Since such a truncated flow is feasible, our ultimate goal is to prove that the truncation step decreases the initial flow only by a constant fraction in expectation.

Let $F(B,C)$ be the random variable denoting the flow on link (B,C); clearly, $F(B,C) = g_{C'}(B,C)$ with probability $p(B,C')$ and its expectation

$$\mathbb{E}[F(B,C)] = f(B,C) \equiv \sum_{C' \in \mathcal{C}} f_{C'}(B,C) = \sum_{C' \in \mathcal{C}} p(B,C') g_{C'}(B,C).$$

Constraint (8) states that the expected inflow $\sum_{B \in \mathcal{B}} f(B,C)$ at client C is at most its demand $\alpha(C)$. Also, for a given C, the $F(B,C)$ values are independent. Finally, constraint (9) enforces that $F(B,C) \leq \alpha(C)$ irrespective of the choice of the assigned capacity at base-station B, (i.e. inflow due to a single base-station at a client never exceeds the demand of the client). Thus, we ensure that there is some restriction on the wasted capacity, i.e. the capacity in the base-client links which are left unused due to truncation; such a restriction was absent in the previous formulation and, as we will see, this additional condition will be sufficient for our purpose.

Note. The rounding procedure can be simplified in an actual implementation. Once we obtain the assigned capacities of all the base-stations using randomized rounding as described above, we can run a maximum flow algorithm on the augmentation graph. Note that this achieves at least as much (and potentially more) flow as that achieved by the rounding procedure described above. So an actual implementation of our algorithm will rather employ a maximum flow sub-routine than the above procedure for determining flows. However, we assume that our algorithm uses the above procedure since it would be simpler to analyze—all bounds proved using this assumption hold for an actual implementation using maximum flow as well.

Before moving on to the analysis of the algorithm, let us verify that the new ILP does represent the original problem. To do this, let us fix some optimal solution (τ^*, f^*) for the original problem. Consider now a solution to our ILP defined as follows. For each base-station B we set $p(B,C) = 1$ if B chooses $\eta(C)$ as its assigned capacity i.e. if $\tau^*(B) = \eta(B,C)$; otherwise $p(B,C) = 0$. Next, for each link (B,C), we set $f_{C'}(B,C) = f^*(B,C)$ if $\tau^*(B) = \eta(B,C')$, and $f_{C'}(B,C) = 0$ otherwise. Observe that all the constraints are preserved, and the objective value corresponding to this solution has value equal to that for the optimal solution. The converse direction is similar and we omit it for brevity.

2.3 Analysis

If there are n base-stations and m clients, then the algorithm clearly runs in time polynomial in $N = \max(n,m)$. So, we focus on proving guarantees on the approximation factor of the algorithm. By the discussion in the previous section, we know that in our rounding procedure the difference between the objective value of the solution to the FLP and the actual flow that we obtain, consists solely of the amount of initial flow that we have to truncate due to overflows at clients. Thus our main task is to prove upper bounds on the expected overflow. Let $F(C) \equiv \sum_{B \in \mathcal{B}} F(B,C)$ be the random variable denoting total inflow at C before truncation and $T(C) \equiv \min(F(C), \alpha(C))$ be the random variable representing the inflow at C after truncation. We would like to show that

$$\mathbb{E}[T(C)] \geq (1-1/e)\mathbb{E}[F(C)]. \tag{13}$$

Then,

$$\mathbb{E}[\sum_{C\in\mathcal{C}} T(C)] \geq (1-1/e)\mathbb{E}[\sum_{C\in\mathcal{C}} F(C)] \geq (1-1/e)T^*, \tag{14}$$

where T^* is the total flow in an optimal integral solution. This proves Theorem 2, which was stated in Section 1.

To establish inequality (13), we will need the following theorem (a similar proof appears in [1]).

Theorem 3. *Suppose we have a sequence of independent discrete random variables* $X_1, X_2, \ldots, X_n$ *such that each* $0 \leq X_i \leq 1$*. Furthermore, suppose* $X = \sum_{i=1}^n X_i$ *and* $\mathbb{E}[X] \leq 1$*. If* $Y = \min(X, 1)$*, then*

$$\mathbb{E}[Y] \geq (1-1/e)\mathbb{E}[X].$$

We first use this theorem to prove inequality (13), and then give a proof of the theorem itself. If, for client C, we define $X_i = F(B_i, C)/\alpha(C)$ (where $\mathcal{B} = \{B_1, \ldots, B_n\}$) and $Y = T(C)/\alpha(C)$, then such X_is and Y satisfy the assumptions of the theorem. Thus, we can conclude that

$$\mathbb{E}[T(C)] = \alpha(C)\mathbb{E}[Y] \geq (1-1/e)\alpha(C)\mathbb{E}[X] = (1-1/e)\mathbb{E}[F(C)].$$

Proof (Theorem 3). Our proof has the following outline. We assume for the sake of contradiction that there exists a sequence $\{\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_n\}$ of discrete random variables such that

$$\mathbb{E}[\hat{Y}] = \mathbb{E}[\min(\sum_i \hat{X}_i, 1)] < (1-1/e)\mathbb{E}[\hat{X}] = (1-1/e)\mathbb{E}[\sum_i \hat{X}_i].$$

We call such a sequence $(\hat{X}_i)$ a *nemesis sequence*. First, we prove that we can assume without loss of generality, that $\hat{X}_i$s are 0-1 random variables. Then, we prove our theorem for 0-1 random variables, thus arriving at a contradiction for the general case.

Let $S(\hat{X}_i)$ be the number of distinct values other than 0 and 1 for which $\hat{X}_i$ has non-zero probability. Now, let us consider a nemesis sequence $(\hat{X}_i)$ that minimizes $\sum_i S(\hat{X}_i)$. We will prove that if $\sum_i S(\hat{X}_i) > 0$ then there exists another nemesis sequence $(\widetilde{X}_i)$ with $\sum_i S(\widetilde{X}_i) < \sum_i S(\hat{X}_i)$. The minimality of $(\hat{X}_i)$ implies there exists a nemesis sequence with $\sum_i S(\hat{X}_i) = 0$, i.e. $(\hat{X}_i)$ is a sequence of 0-1 variables.

If $\sum_i S(\hat{X}_i) > 0$, then there exists some k such that $S(\hat{X}_k) > 0$, which in turn means that there exists some $0 < a < 1$ such that $Pr[\hat{X}_k = a] = p > 0$. Suppose that this $\hat{X}_k$ takes value of 0 and 1 with probability $q \geq 0$ and $r \geq 0$ respectively (note that p, q and r do not necessarily sum to 1). Now, consider another random variable $\widetilde{X}_k$ that is distributed identically to $\hat{X}_k$ except that the probabilities of a, 0 and 1 are changed to 0, $q+(1-a)p$ and $r+ap$ respectively. Note that $\mathbb{E}[\widetilde{X}_k] = \mathbb{E}[\hat{X}_k]$, $0 \le \widetilde{X}_k \le 1$ and $S(\widetilde{X}_k) = S(\hat{X}_k) - 1$. So, if we define $\widetilde{X}_i = \hat{X}_i$ for $i \neq k$, then $\mathbb{E}[\hat{X}] = \mathbb{E}[\widetilde{X}] \le 1$. We would like to compare $\mathbb{E}[\hat{Y}]$ to $\mathbb{E}[\widetilde{Y}] \equiv \mathbb{E}[\min\{\widetilde{X}, 1\}]$. Note that by our definition, for any $\delta \ge 0$,

$$Pr[\widetilde{X} - \widetilde{X}_k = \delta] = Pr[\hat{X} - \hat{X}_k = \delta].$$

Thus, to prove that $\mathbb{E}[\widetilde{Y}] \le \mathbb{E}[\hat{Y}]$, it is sufficient to prove that

$$\mathbb{E}[\widetilde{Y} | \widetilde{X} - \widetilde{X}_k = \delta] \le \mathbb{E}[\hat{Y} | \hat{X} - \hat{X}_k = \delta],$$

for all $\delta \ge 0$.

Clearly, if $\delta \ge 1$ then

$$\mathbb{E}[\widetilde{Y} | \widetilde{X} - \widetilde{X}_k = \delta] = 1 = \mathbb{E}[\hat{Y} | \hat{X} - \hat{X}_k = \delta];$$

so the inequality holds. On the other hand, for $\delta < 1$,

$$\begin{aligned} \mathbb{E}[\widetilde{Y} | \widetilde{X} - \widetilde{X}_k = \delta] - \mathbb{E}[\hat{Y} | \hat{X} - \hat{X}_k = \delta] &= \mathbb{E}[\min\{\widetilde{X}_k, 1-\delta\}] - \mathbb{E}[\min\{\hat{X}_k, 1-\delta\}] \\ &= ap(1-\delta) - p\min\{a, 1-\delta\} \le 0. \end{aligned}$$

Thus, $\mathbb{E}[\widetilde{Y}] \le \mathbb{E}[\hat{Y}]$, which proves that $\{\hat{X}_i\}$ had to be a zero-one nemesis sequence.

Now, when $\{\hat{X}_i\}$ is zero-one,

$$\begin{aligned} \mathbb{E}[\hat{Y}] &= Pr[\hat{X} \ge 1] \\ &= 1 - \prod_i (1 - Pr[\hat{X}_i = 1]) \\ &\ge 1 - (1 - \sum_i \mathbb{E}[\hat{X}_i]/n)^n \\ &\ge 1 - e^{-\mathbb{E}[\hat{X}]} \\ &\ge (1 - 1/e)\mathbb{E}[\hat{X}], \end{aligned}$$

as desired, where in the first inequality we used the fact that

$$\sum_i Pr[\hat{X}_i = 1] = \sum_i \mathbb{E}[\hat{X}_i] = \mathbb{E}[\hat{X}],$$

and the arithmetic/geometric mean inequality.

To conclude, we may note that this theorem is tight for n i.i.d. 0-1 random variables X_i with $Pr[X_i = 1] = 1/n$.

3 Conclusion

The ability to adaptively change channel widths in wireless networks introduces interesting algorithmic problems. In this paper, we have studied a throughput maximization problem in infrastructure wireless networks that was identical to the maximum flow problem in bipartite graphs with adaptive channel width. An interesting open question is to to find maximum flow in a *general* network with adaptive channels.

4 Acknowledgements

We thank the anonymous referees for their helpful comments.

References

1. N. Andelman and Y. Mansour. Auctions with budget constraints. In *In 9th Scandinavian Workshop on Algorithm Theory*, pages 26–38, 2004.
2. Y. Azar and O. Regev. Combinatorial algorithms for the unsplittable flow problem. *Algorithmica*, 44(1):49–66, 2006.
3. A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47(1):53–78, 2007.
4. R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A case for adapting channel width in wireless networks. In *SIGCOMM*, pages 135–146, 2008.
5. C. Chekuri, S. Khanna, and F. B. Shepherd. An O(sqrt(n)) approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2(1):137–146, 2006.
6. E. A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Doklady (Doklady)*, 11:1277–1280, 1970.
7. J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.
8. S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM J. Comput.*, 4(4):507—518, 1975.
9. L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.
10. A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783—797, 1998.
11. A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921—940, 1988.
12. V. Guruswami, S. Khanna, R. Rajaraman, F. B. Shepherd, and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *J. Comput. Syst. Sci.*, 67(3):473–496, 2003.
13. J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.
14. D. R. Karger. Using random sampling to find maximum flows in uncapacitated undirected graphs. In *STOC*, pages 240—249, 1997.
15. D. R. Karger. Better random sampling algorithms for flows in undirected graphs. In *SODA*, pages 490–499, 1998.
16. D. R. Karger and M. S. Levine. Finding maximum flows in undirected graphs seems easier than bipartite matching. In *STOC*, pages 69—78, 1998.
17. D. R. Karger and M. S. Levine. Random sampling in residual graphs. In *STOC*, pages 63–66, 2002.
18. J. M. Kleinberg. Single-source unsplittable flow. In *FOCS*, pages 68–77, 1996.
19. T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan. Load-Aware Spectrum Distribution in Wireless LANs. In *ICNP*, 2008.
20. C. E. Shannon. Communication in the presence of noise. *Proc. Institute of Radio Engineers*, 37(1):1021, 1949.
21. A. Srinivasan. Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems. In *FOCS*, pages 416–425, 1997.